贪心算法与动态规划的比较

贪心策略要点

- 贪心算法是通过做一系列选择来获得最优解，在算法里的每一个决策点上，都力图选择最好的方案，这种启发式策略并非总能产生最优解

- 之前介绍的贪心算法的步骤
  - 确定问题的最优子结构
  - 给出递归解(指递归方程)
  - 证明在递归的每一步，有一个最优的选择是贪心的选择，因此做出这种选择是安全的。
  - 证明除了贪心选择导出的子问题外，其余子问题都是空集合
  - 根据贪心策略写出递归算法
  - 将递归算法转换为迭代算法

贪心策略要点

上述步骤是以动态规划为基础的。实际上可改进它，重点关注**贪心选择**。例如，活动选择问题可改进为(直接写出递归解)：

- 首先定义子问题 S_{ij}, i, j 均可变
- 如果我们总是做贪心选择，则 $S_{ij} \Rightarrow S_{i,n+1}$

  ∵$n+1$不变，**可省略**，子问题变为：

  $S_i = \{a_k \in S: f_i \leq s_k\}$ // S_i表示a_i完成后开始的任务集
- 证明一个贪心的选择(即选 S_i中第一个完成的活动a_m)，和剩余子问题S_m (与a_m相容)的最优解**结合**，能产生S_i的**最优解**。

贪心策略要点

- 贪心算法的一般设计步骤

  ① 将优化问题分解为做出一种选择及留下一个待解的子问题

  ② 证明对于原问题总是存在一个最优解会做出贪心选择，从而保证贪心选择是安全的

  ③ 验证当做出贪心选择之后，它和剩余的一个子问题的最优解组合在一起，构成了原问题的最优解

- 没有一个一般的方法告诉我们一个贪心算法是否会解决一个特殊的最优问题，但是有两个要素有助于使用贪心算法：

  贪心选择性质，最优子结构

贪心策略要点

1、贪心选择性质

- **贪心选择性质**：一个全局最优解能够通过局部最优(贪心)选择达到
- 贪心法总是在当前步骤上选择最优决策，然后解由此产生的子问题
- 贪心选择只依赖了目前所做的选择，但不依赖于将来的选择及子问题的解
- 自顶向下，每做一次贪心选择，就将子问题变得更小
- **贪心算法一般总存在相应的动态规划解**，但贪心法的效率更高，原因：

  ①对输入做预处理

  ②使用合适的数据结构(如优先队列)

贪心策略要点

2、最优子结构

- 和动态规划一样，该性质也是贪心法的一个关键要素

  例如，活动选择问题的动态规划解中：

  S_{ij}的最优解A_{ij} ➔若$a_k \in A_{ij}$，则A_{ij}包含S_{ik}和S_{kj}的最优解

- 对贪心算法更直接

  原问题的最优解➔贪心选择+子问题的最优解

  通常可使用归纳法证明在每一步上做贪心选择可产生最优解，

  由此可导出最优子结构

贪心策略要点

3、贪心法与动态规划的比较

- 相同点：两种方法都利用了最优子结构特征
- 易错误处：
  - ① 当贪心算法满足全局最优时，可能我们试图使用动态规划求解，但前者更有效
  - ② 当事实上只有动态规划才能求解时，错误地使用了贪心法

为了说明两种技术的细微区别，请看一个古典优化问题的两个变种：

背包问题：n个物品重$w_1, w_2, \ldots, w_n$，背包可放入重量W，问能否从n件物品中选择若干件放入背包，使重量和正好等于W。

贪心策略要点

- 0-1背包问题和分数背包问题

  物品件数：n；第i件物品价值$\$v_i$，重量$w_i$磅，

  v_i和w_i为整数，背包载重量：W磅(整数)

  怎样选物品装包使得包内含有价值最大，且总重量$\leq W$

  不允许一物品多次装包，可以理解为 n 个物品互不相同

  - **0-1背包：某物品拿与不拿(1,0的选择)**
  - **分数背包：某物品可取部分装包**

  想象：公司搞活动，包容量有限，员工都想尽可能使装走的东西总价值最大，0-1背包装的只能是**金条**之类物品，分数背包装的可以是**金粉**之类物品。

贪心策略要点

- **两个背包问题都具有最优子结构!!**
  - **0-1背包问题 (0-1 knapsack problem)**

    原问题的最优解：设背包中装有重量至多为W磅的最有价值的物品(取自n件物品)；

    子问题的最优解：从原问题的最优解中去掉某物品 j，则包中剩余物品应是除 j 外，取自原$n-1$件物品中最有价值，且总重不超过$W-w_j$的若干物品。

    显然，原问题分解为子问题时，原问题的最优解也要求子问题的解最优

贪心策略要点

- 分数背包 (fractional knapsack problem)

  子问题：从背包中去掉物品 j (Note：part of)，重量为w(该物品剩余 $w_j - w$)，则包中剩余物品应是除 j 之外，取自原$n-1$件物品以及物品 j 的 $w_j - w$ 部分中最有价值且总重至多为 $W - w$ 的若干物品

  显然，原问题解最优亦要求子问题的解最优

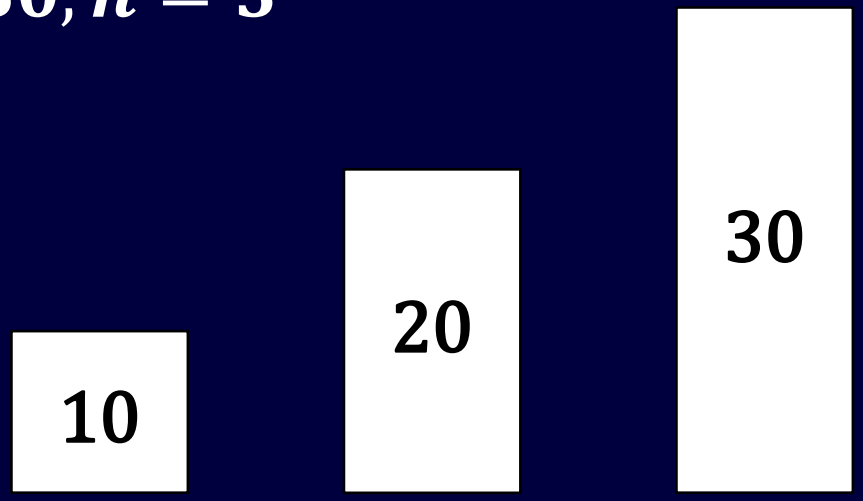
贪心策略要点

- 两个背包问题的不同解法
  - 求解0-1背包<u>最优解</u>只能用动态规划求解
    - 按每磅价值 (v_i/w_i) 排序
    - 贪心选择策略：取单位价值最大者装包，若装不下，考虑下一单位价值最大的物品，直至包装满或所有物品都考虑过为止
    - 实际上，装入当前每磅价值最大者只能保证当前最优(局部最优)，然而放弃它可能使得后续选择更优。所以在装包前，应将某物品装包的子问题的解和放弃它的子问题的解进行比较，这将导致许多重叠子问题，这正是动态规划的特点。

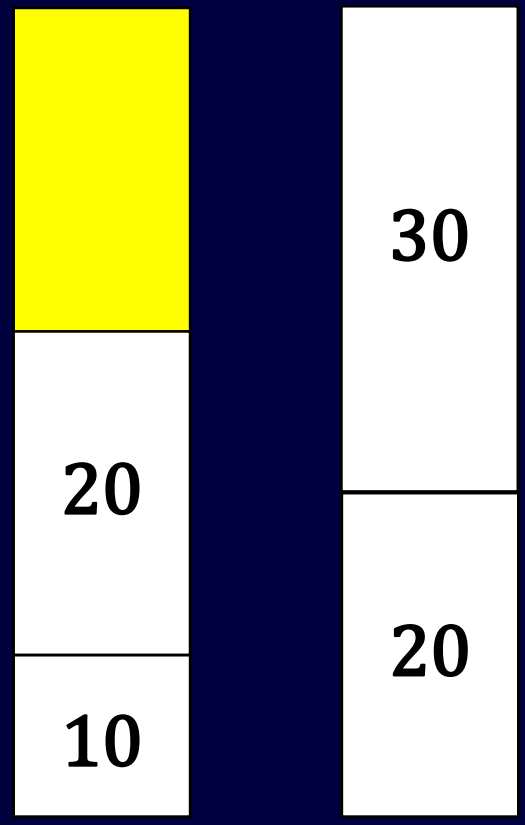
贪心策略要点

- 0-1背包：只能用**动态规划**求出最优解

$W = 50, n = 3$



V_i:	\$60	\$100	\$120
V_i/W_i:	6	5	4



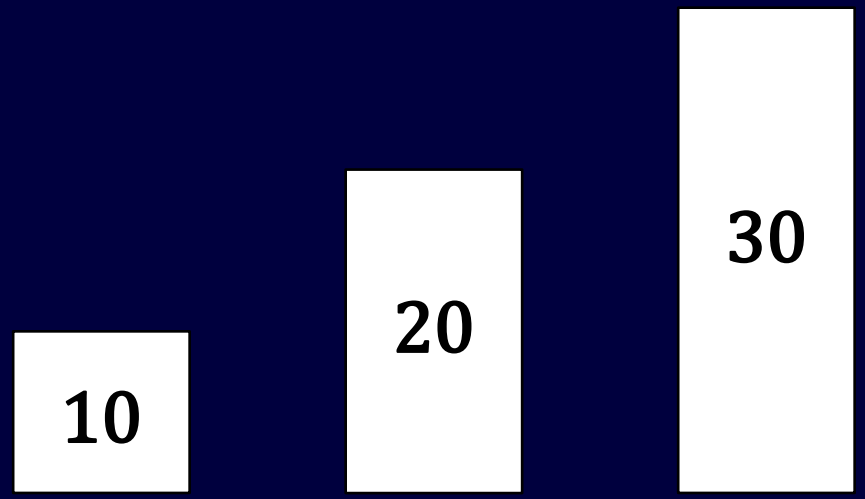
装包方案

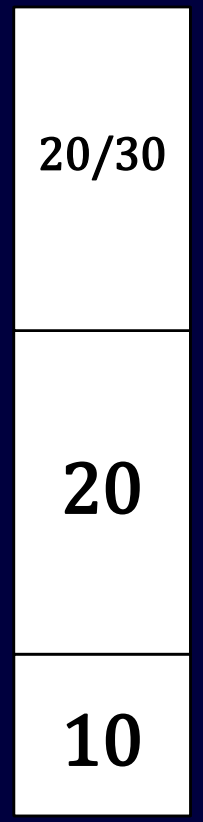
\$160	\$220	总价值
贪心解	≠	最优解

贪心策略要点

- 分数背包：可用贪心算法求出最优解

$W = 50, n = 3$





装包方案

$240 $240 总价值

贪心解 = 最优解